



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/650,257	08/27/2003	Bhavesh P. Davda	4366-120	7452
48500	7590	04/09/2007	EXAMINER	
SHERIDAN ROSS P.C.			WEI, ZHENG	
1560 BROADWAY, SUITE 1200				
DENVER, CO 80202				
			ART UNIT	PAPER NUMBER
			2192	
SHORTENED STATUTORY PERIOD OF RESPONSE		MAIL DATE	DELIVERY MODE	
3 MONTHS		04/09/2007	PAPER	

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No. 10/650,257	Applicant(s) DAVDA, BHAVESH P.	
	Examiner Zheng Wei	Art Unit 2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 16 January 2007.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-21 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-21 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 16 January 2007 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

Remarks

1. This office action is in response to the amendment filed on 01/16/2007.
2. Claims 1, 6, 8, 10, 11, 14 and 15 have been amended.
3. The objections of drawings file on 09/29/2003 are withdrawn in view of the Applicant's new drawings filed on 01/16/2007.
4. The objections of claims 8 and 15 are withdrawn in view of Applicant's amendment.
5. The 35 U.S.C. 112 first paragraph rejection of claims 14 and 15 are withdrawn in view of the Applicant's amendment.
6. The 35 U.S.C. 112 second paragraph rejection of claims 1-9 are withdrawn in view of the Applicant's amendment.
7. The 35 U.S.C. 101 rejection of claims 10-17 are withdrawn in view of the Applicant's amendment
8. Claims 1-21 remain pending and have been examined.

Claim Rejections - 35 USC § 102

9. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

10. Claims 1-5, 7, 10-13 and 16-19 are rejected under 35 U.S.C. 102(b) as being anticipated by Lillich (Lillich et al., US 5,619,698)

Claim 1:

Lillich discloses a method for updating a running process, comprising:

- allocating in executable program code text first memory space operable to receive new program instructions comprising at least a first updated function (See for example, Fig.2, element 130, 132 and related text);
- allocating in executable program code text second memory space operable to receive address information related to said new program instructions (See for example, Fig.2, element 110 and related text, also see col.2, lines 19-20, "a trap table resident in RAM");
- running said executable program code (See for example, col.2, lines 48-50, "During execution of application code 102...");
- stopping execution of said executable program code (See for example, col.2, lines 49-50, "the 68K micro-processor will encounter and execute the ATRAP 104");
- injecting a jump instruction and an address of an update table at a location in a memory containing a first instruction of a first replaced function, wherein said address of said update table contains an address of a first instruction of said first updated function (See for example, col.3, lines 1-5, "the ATRAP instruction 104" (injected jump instruction) and "trap table 110" (contains address to the corresponding system routine) and also Fig.2, Fig.6 teach the art (see for example, Fig.6, element 465 "re-direct code" (jump instruction) and element 469-473, "pointer" (address of update table))); and
- resuming execution of said executable program code, wherein said first updated function is called in place of said first function, and wherein said executable code is updated in said memory (See for example, Fig.2, element 150 and related text, also see col.3, lines 1-5, "and then jumps to the corresponding system routine.")

Claim 2:

Lillich further discloses the method of Claim 1, wherein said step of resuming execution of said executable program code comprises running an intermediate executable, wherein said intermediate executable comprises said updated copy of said executable program code in said memory (See for example, col.3, lines 45-51, "address_1 will point to patch code 132 located in RAM 130 rather than the original system routine").

Claim 3:

Lillich also discloses the method of Claim 1, further comprising: before said step of running said executable program code, copying said executable program code to said memory (See for example, Fig.2, element 122 and related text, "system code").

Claim 4:

Lillich also discloses the method of Claim 1, further comprising:

- injecting a jump instruction and an address of said update table at a location in a stored copy of said executable program code containing an address of said first function (See for example, col.3, lines 1-5, "the trap dispatcher 108"); and
- populating said update table with an address of said first updated function, wherein a stored copy of said executable program code is updated (See for example, Fig.2, element 110, "trap table" and related text).

Claim 5:

Lillich further discloses the method of Claim 4, wherein said updated stored copy of said executable program code comprises final updated executable program code (See for example, Fig.2, elements 146, 150, 152 and related text in col.3, lines 45-51")

Claim 7:

Lillich further discloses the method of Claim 1, wherein said injected jump instruction replaces a first instruction of said first replaced function (See for example, col.3, lines 1-5, "the ATRAP instruction 104").

Claim 10:

Lillich discloses a computer implemented method, the method comprising:

- receiving information identifying: a running executable program to be patched and a function to be replaced (See for example, col.8, line 67-col.9, line 8, "TVector");
- accessing a symbol table in a memory for said executable program to be patched (See for example, col.9, lines 9-13, "binding manager");
- obtaining from said symbol table an address of said function to be replaced (See for example, col.9, lines 10-12, "the address of the code or data");
- stopping execution by a processor of said running executable program to be patched (See for example, fig.14, steps 852 and related text);
- injecting in said running executable program to be patched at a location in said memory containing a first instruction of said function to be replaced a jump instruction and an address of a new function, wherein said new function is executed in place of said function to be replaced, and wherein a patched version of said executable program is created in memory (See for example, col.19, lines 29-40, "the address of the desired extension patch" and also see col.3, lines 1-5, "the ATRAP instruction 104" (injected jump instruction) and "trap table 110" (contains address to the corresponding system routine). The Fig.2 and Fig.6 also teach the art (see for example, Fig.6, element 465 "re-direct code" (jump instruction) and element 469-473, "pointer" (address of update table))) and

- resuming execution of said executable program by said processor, wherein said patched version of said executable program is executed by said processor (See for example, col.19, lines 41-46, "trap dispatcher").

Claim 11:

Lillich discloses the method of Claim 10, further comprising providing a jump table in said memory, wherein said injecting in said running executable program to be patched a jump instruction and an address of a new function comprises replacing a first instruction line associated with said function to be replaced with an instruction to jump to a first address contained within said jump table, and wherein said first address contained within said jump table comprises an address of a first instruction line of said new function (See for example, Fig.2, element 110 and related text, "trap table").

Claim 12:

Lillich also discloses the method of Claim 11, wherein said first instruction line associated with said function to be replaced comprises a first instruction of said function to be replaced, wherein said first instruction is not an instruction to jump to another address, and wherein said first instruction of said function to be replaced is replaced by said instruction to jump to a first address (See for example, Fig5 and fig.7, element 560, 400 and related text).

Claim 13:

Lillich further discloses the method of Claim 11, wherein said first instruction line associated with said function to be replaced comprises an instruction to jump to a second address contained within said jump table, and wherein said instruction to jump to a second address is replaced by said instruction to jump to a first address. (See for example, Fig5 and fig.7, element 560, 400 and related text, "TVector").

Claim 16:

Lillich also discloses the method of claim 10, wherein said computational component comprises a computer readable storage medium containing instructions for performing the method (See for example, p.5, lines 36-38, "A first aspect of the present invention teaches a computer implemented method for integrating patches into a computer operating system").

Claim 17:

Lillich further discloses the method of claim 10, wherein said computational component comprises a logic circuit. (See for example, Fig.1, element 52, "CPU")

Claims 18-19:

Claims 18 and 19 are system claims for updating executable program code using the methods discussed in claims 10-11. Therefore, accordingly these claims would also be anticipated by Lillich.

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

12. Claims 6, 8, 14 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lillich (Lillich et al., US 5,619,698) in view of Buban (Buban et al., US 2004/0107416)

Claim 6:

Lillich discloses the method as in claim 1 above, but does not further disclose the method further comprising:

- determining a first distance between a position within said code text at which execution of said executable program code is stopped and an address of a first function, wherein said first function is a function to be updated; and
- in response to said first distance exceeding a predetermined amount, populating an update table stored in memory with an address of a first updated function

However, Buban in the same analogous art of patching of in-use functions on a running computer system discloses a method to determine a distance between two memory addresses (p.5, paragraph[0047], "move backward or forward").

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use this method to determine the distance between those two addresses. One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine as once suggested by Buban (See, p.5, paragraph[0046], "atomically updated").

Claim 8:

Lillich and Buban disclose the method as in claim 6 above, but Lillich does not disclose the predetermined amount about the distance. However, Buban further discloses wherein said predetermined amount is 8 bytes. (See, p.5, paragraph[0046], "a 64-bit (8-byte) word on an Intel x86 processor"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the

invention was made to use this predetermined amount 8 byte. One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine if the processor's smallest unit of atomically replaceable memory is 8 byte (See, p.5, paragraph[0046], "atomically updated").

Claim 14:

Lillich discloses the computational component for performing the method of claim 10, but does not disclose the method further comprising:

- determining a number of bytes between a location within said executable program at which said executable program is stopped and an address of said function to be replaced.

however, Buban in the same analogous art of patching of in-use functions on a running computer system discloses a method to determine a distance between two memory addresses (p.5, paragraph[0047], "move backward or forward").

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use this method to determine the distance between those two address. One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine (See, p.5, paragraph[0046], "atomically updated").

Claim 15:

Lillich and Buban disclose the method of Claim 14, Lillich also discloses in response to said determined number of bytes being at least as great as a first selected number, injecting in a stored copy of said running executable program to be patched said jump instruction and said address of said new function in place of said address of said function to be replaced, wherein a patched version of said executable program is created, but does not disclose the method of injecting is based on selected number. However Buban discloses the reason in response to

Art Unit: 2192

said determined number of instructions being at least as great as a first selected number (See, p.5, paragraph[0046], "A processor's smallest unit of atomically replaceable memory, e.g., a 64-bit (8-byte) word on an Intel x86 processor"). One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine (See, p.5, paragraph[0046], "atomically updated").

13. Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lillich (Lillich et al., US 5,619,698) in view of Munsil (Munsil et al, US 2003/0167463)

Claim 9:

Lillich discloses the method of Claim 1, but does not disclose wherein said second memory space is read only memory space. However, Munsil discloses a method for applying custom application-compatibility fix using read only memory (see for example, p.4 paragraph[0033], "using read-only memory to prevent modification and corruption by unauthorized or unknowledgeable parties").

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use read only memory to store patch address information. One would have been motivated to do so to protect important address information.

14. Claims 20 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hundt (Hundt et al., US 2002/0152455) in view of Buban (Buban et al., US 2004/0107416)

Claim 20:

Hundt discloses a system for dynamic instrumentation of an executable program, comprising:

- a create patch tool operable to receive information identifying a executable program to be updated and a function to be replaced (See for example, Fig.1, steps 102-108 and related text);
- a patch tool operable to query an operating system for a process identifier associated with said identified executable program (See for example, fig.1, step 104 and related text, also see p.2, paragraph[0030], "an available thread is selected from the application");
- a debugging utility operable to stop execution of said executable program to be updated and to determine a position of an instruction pointer associated with said executable program to be updated (See for example, Fig.1, step 110 and related text. "patch function entry points with breakpoints"); and
- a signal handler tool operable to replace in memory an address of said function to be replaced with an address of a replacement function in response to said position of said instruction pointer being at least a predetermined distance from said address of said replacement function, wherein said replacement function is executed instead of said function to be replaced upon resuming execution of said executable program, wherein said executable program is updated. (See for example, paragraph[0018], "executing the instrumented functions instead of the original functions")

but does not disclose replace memory address according to the predetermined distance. However, Buban in the same analogous art of patching of in-use functions on a running computer system discloses a method to determine a distance between two memory addresses (p.5, paragraph[0047], "move backward or forward"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use this method to determine the distance between those two address. One would have been motivated to perform atomic upgrading so that no processor that might execute the subroutine will see a partially updated version of the patched routine (See, p.5, paragraph[0046], "atomically updated").

Claim 21:

Hundt and Buban disclose the system of Claim 20 above, Hundt further discloses, the signal handler is additionally operable to replace in a stored copy of said executable program an address of said function to be replaced with an address of a replacement function, wherein said stored copy of said executable program is updated (See for example, Fig.1, step 124 and related text).

Response to Arguments

15. Applicant's arguments filed on 01/16/2007, in particular on pages 10-13, has been fully considered but they are not persuasive. For example:

At pages 11-13, Applicant contend s that claims 1, 10 and 18 are not anticipated by Lillich, as Lillich does not disclose followings:

- Claim 1, Lillich does not discloses "injecting a jump instruction and an address of an update table at a location in a memory containing a first instruction of a first replaced function, wherein said address of said update table contains an address of a first instruction of said first updated function"
- Claim 10, Lillich does not discloses "injecting in said running executable program to be patched at allocation in said memory containing a first instruction of said function to be replaced a jump instruction and an address of a new function, wherein said new function is executed by said processor in place of said function to be replaced, and wherein a patched version of said executable program is created in said memory"

Art Unit: 2192

- Claim 18, Lillich does not disclose “means for inserting a jump code and an address associated with a replacement function in place of an address of said function to be replaced in said existing executable program code, wherein updated executable program code is created”

However, the Examiner strongly disagrees. As in the previous Office Action, at column 3, lines 1-3, “ATRAP instruction 104” (injected jump instruction) and “trap table 110” (contains address to the corresponding system routine) and also Fig.2, Fig.6 teach the art (see for example, Fig.6, element 465 “re-direct code” (jump instruction) and element 469-473, “pointer” (address of update table)). Therefore, Lillich does teach all the limitations for claims 1, 10 and 18.

At page 14, the Applicant asserts that cited references Hyndt and Buban do not teach the limitation of claims 20 and 21 wherein the system includes a signal handler tool. However, Hunt in Fig.2A, steps 208-210 disclose, “overwrite code in application executable with code to allocate and map shared memory (injected code)” and “Initialize register states for the selected thread, including the address of the injected code as the program counter for the thread”. Therefore, it is obvious that the functions address above are performed or realized by tool/program like the “signal handler tool”.

Conclusion

16. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.
17. Applicant's arguments have been fully considered but they are not persuasive and accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a).

Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than **SIX MONTHS** from the mailing date of this final action.

18. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Zheng Wei whose telephone number is (571) 270-1059 and Fax number is (571) 270-02059. The examiner can normally be reached on Monday-Thursday 8:00-15:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The

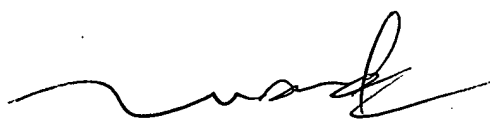
Art Unit: 2192

fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571- 272-1000.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

ZW



TUAN DAM
SUPERVISORY PATENT EXAMINER